PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : G06F 9/46

A1

(11) International Publication Number: **WO 00/39678**

(43) International Publication Date: 6 July 2000 (06.07.00)

(21) International Application Number: PCT/US99/30827

(22) International Filing Date: 27 December 1999 (27.12.99)

(30) Priority Data:

60/114,099	29 December 1998 (29.12.98)	US
09/252,391	18 February 1999 (18.02.99)	US
09/390,928	7 September 1999 (07.09.99)	US

(81) Designated States: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(71) Applicant: CITRIX SYSTEMS, INC. [US/US]; 6400 N.W. 6th Way, Fort Lauderdale, FL 33309 (US).

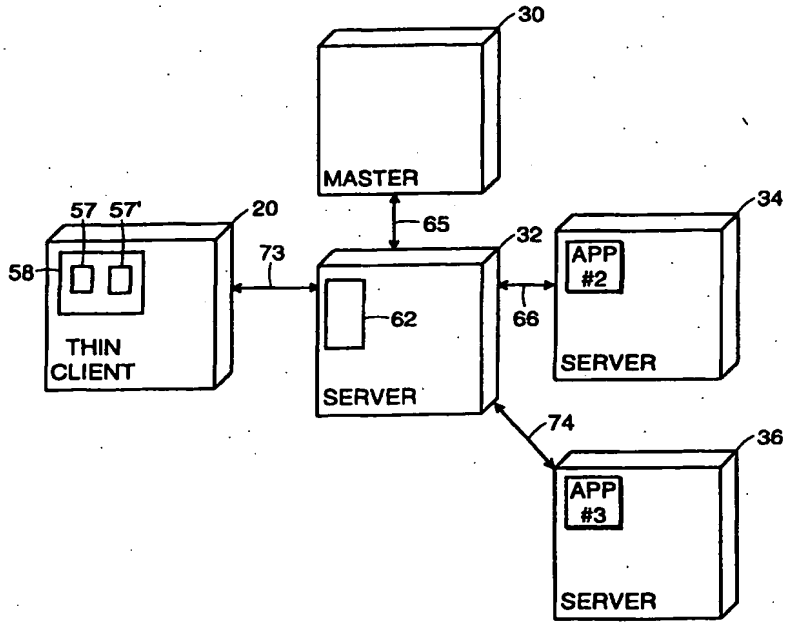
(72) Inventors: DUURSMA, Martin; 4 Orchid Place, West Pennant Hills, NSW 2125 (AU). PANASYUK, Anatoly; 6/34 Forster Street, West Ryde, NSW 2114 (AU). UNGERMAN, Anthony; 18039 Jazz Lane, Boca Raton, FL 33496–0000 (US). PEDERSEN, Bradley, Jay; 7700 S. Woodridge Drive, Parkland, FL 33067 (US). DAVIS, Tom, C., III; 10109 Northwest 17th Street, Coral Springs, FL 33071–5808 (US). BLOOMFIELD, Marc, A.; 2749 S.E. 11th Street, Pompano Beach, FL 33062 (US). CIRALDO, Robert; 1354 Bayview Drive, Apt. K, Fort Lauderdale, FL 33304 (US).

(74) Agent: LANZA, John, D.; Testa, Hurwitz & Thibeault, LLP, High Street Tower, 125 High Street, Boston, MA 02110 (US).

(54) Title: AN APPARATUS AND METHOD FOR DETERMINING A PROGRAM NEIGHBORHOOD FOR A CLIENT NODE IN A CLIENT–SERVER NETWORK

(57) Abstract

An apparatus and method for determining a program neighborhood of a client node in a client–server network is described. The program neighborhood of the client node includes application programs hosted by application servers on the network. The present invention enables a user of a client node to learn of these application programs. The user is not required to know where to find such applications or to manually establish links to such applications. To make the client node aware of its program neighborhood, a host server collects application–related information corresponding to application programs hosted by the servers in the network. The application–related information can include the application name, the server location of the application, minimum capabilities required of client nodes for executing the application, and those users who are authorized to use that application. User credentials are received from the client system. The user credentials are used to filter the application–related information. Information representing those application programs that are available to the client node is transmitted from the host server to the client system for display. In a Windows–based client node, the information can be represented by a graphical icon for each available application program.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

AN APPARATUS AND METHOD FOR DETERMINING A PROGRAM NEIGHBORHOOD FOR A CLIENT NODE IN A CLIENT-SERVER NETWORK

Related Applications

This application claims the benefit of U.S. Provisional Application, Serial No. 60/114,099 filed December 29, 1998 and is a continuation-in-part of United States Patent Application Serial No. 09/252,391, filed February 18, 1999.

Field of the Invention

The invention relates generally to client-server networks. More specifically, the invention relates to a method for informing a client system of application programs installed at server systems that are available to the client system for execution.

Background of the Invention

Contemporary computer networks consist of a number of computer systems, called nodes, communicating with other computer systems via communication links. Typically, some of the nodes are client nodes and other nodes are server nodes. A client node formulates and delivers queries to a server node. A user of the client node enters the queries through a user interface operating on the client node. The server node evaluates the queries and delivers responses to the client node for display on the client user interface.

Usually, the server nodes host a variety of application programs that can be accessed and executed by client nodes. When a client node launches an application program, the execution of that application program can occur at either the client node or the server node, depending upon the computing model followed by the computer network. In a server-based computing model, the server node executes the application program, and only the control information for the client

user interface is transmitted across the computer network to the client node for display. In a client-based computing model, the server node transmits the application program to the client node so that the client node can run the program using the resources of the client node.

One drawback of contemporary computer networks is that client nodes may be unaware of the application programs available for use on the server nodes. In fact, client nodes may not even be aware of each available server node on the network. To find available application programs on a particular server node, a user of the client node may need to find and gain access to that server node and perform a directory listing of the files existing on that server node. Even then, this listing might not indicate to the user those applications which the user is authorized to use.

Moreover, once the user is aware of the application programs on a server node, often that user must establish a link to those applications. Industry has developed software tools to aid the user in creating these links, e.g., Remote Application Manager manufactured by Citrix Systems, Inc., of Ft. Lauderdale, Florida. The Remote Application Manager is an application program installed on the client node. This application program guides an administrator of the client node to enter information that establishes a link to an application published on a particular server node. The administrator supplies such information as the name of the session, the protocol for accessing the server node, the server name or IP address or the published application. Approaches of this kind require that the administrator know the information and understand the details of protocols and domains in order to establish the connection.

Summary of the Invention

The present invention enables the user of a client system to become informed about available application programs on servers in a network without requiring the user to know where to find such applications or to enter technical information necessary to link to such applications.

In one aspect, the invention relates to a method for presenting application programs to a client system in a network including the client system and a plurality of servers. The servers include a host server that receives application-related information corresponding to application programs hosted by a plurality of servers in a network. User credentials are received from the client system. Whether each hosted application program is available to the client system for execution is determined based on the user credentials and the received application-related information. Information is transmitted from the host server to the client system indicating to the client system each hosted application that is available to the client system for execution. In some embodiments, the host server creates an output display containing the application-related information, such as a HyperText Markup Language (HTML) file, an Extended Markup Language (XML), or other Structured General Markup Language (SGML) file.

The host server can authenticate the client system based on received user credentials. When the client system selects one of the available applications for execution, that application can be executed without requiring additional input of user credentials by a user of the client system, although the selected application is at a server other than the host server.

In one embodiment, a connection is established between the client system and the host server using an Independent Computing Architecture (ICA) protocol. The ICA protocol can establish a virtual channel to transmit information to the client system indicating each hosted application program that is available to the client system.

A request to execute one of the available hosted application programs can be received on the connection. In response to the request, a second connection between the client system and the host server can be established to exchange information associated with an execution of the requested application program. In another embodiment, a connection between the host server and a second server hosting the requested application can be established in response to the request. Information associated with an execution of the requested application program can be exchanged between the second server and the client system via the host server.

A second request to execute a second one of the available hosted application programs can be received on the connection between the host server and the client system. Another connection between the host server and a third server hosting the requested application can be established in response to the second request. Information associated with an execution of the second requested application program can be exchanged between the third server and the client system via the host server. Information received from the second and third servers can be merged for transmission to the client system.

In still another embodiment, application information can be provided by the host server to the client system in response to the first request. A second connection can be established between the client system and a second server hosting the requested application in response to the provided application information.

The host server can transmit the available application information in response to a request by the client system. The transmitted information can produce a graphical user interface display at the client system having icons representing the available application programs. The transmitted information can also disclose each hosted application program that the client system is unauthorized to use.

The host server can communicate with each of the other servers to determine the application programs hosted by those other servers and to develop a database for storing the application-related information. The database can be accessible to each of the other servers of the plurality, or each of the other servers can maintain a database for storing the application-related information.

In another aspect, the invention relates to a method for presenting to a client system each application hosted by a server that is available to the client system. User credentials are received from the client system. Application-related information is maintained by the server. Whether each application hosted by the server is available to the client system for execution is determined based on the user credentials and the application-related information. Information indicating each application that is available to the client system is transmitted from the server to the client system.

In another aspect, the invention relates to a server comprising a service module that collects application-related information corresponding to application programs hosted by the plurality of servers. A database stores the collected application-related information. A receiver receives user credentials from the client system. The service module determines for each application program hosted by the plurality of servers whether that hosted application program is available for use by the user of the client system based on the user credentials and the application-related information stored in the database. A transmitter transmits information to the client system.

The service module can transmit a datagram to other servers in the plurality to collect the application-related information corresponding to the application programs hosted by those servers. The transmitted information discloses to a user of the client system each hosted

application program that is available for use by the user of the client system. The transmitter can transmit available application information using a virtual channel communications protocol.

Brief Description of the Drawings

The invention is pointed out with particularity in the appended claims. The above and further advantages of the invention may be better understood by referring to the following description in conjunction with the accompanying drawings, in which:

Fig. 1 is a diagram of an embodiment of client nodes in communication with a group of server nodes via a network, wherein a program neighborhood of a client node can be determined according to the principles of the invention;

Fig. 2A is a block diagram illustrating an exemplary process by which one of the server nodes can initiate execution of an application program for determining the program neighborhood of a client node;

Fig. 2B is a block diagram illustrating an exemplary process by which a client node can initiate execution of an application program for determining the program neighborhood of that client node;

Fig. 2C is a block diagram illustrating an exemplary process by which a client node uses a web browser application to determine its program neighborhood;

Figs. 3A, 3B, and 3C are block diagrams illustrating exemplary processes by which a client node can launch an application program from a Program Neighborhood window displayed at that client node;

Figs. 3D is a block diagram illustrating an exemplary process by which a client node can launch an application program from a Program Neighborhood web page displayed at that client node;

Fig. 4 is a block diagram of a client-based computing embodiment in which a client node having an installed program neighborhood application of the invention is in communication with one of the server nodes;

Fig. 5 is a block diagram of a server-based computing embodiment in which a client node is in communication with a server node having an installed program neighborhood application program of the invention;

Fig. 6A is a screen shot of an exemplary display on the display screen of a client node after the program neighborhood application program of the invention is executed;

Fig. 6B is a screen shot of another exemplary display on the display screen of a client node after the program neighborhood application program of the invention is executed;

Fig. 7 is a flow chart representation of an embodiment of a process by which a client node is informed as to the availability for use of application programs on the application servers.

Detailed Description of the Invention

FIG. 1 shows a first computing system (client node) 10 and a second computing system (client node) 20 in communication with computing systems (application servers) 30, 32, 34, and 36 over a network 40. The network 40 can be a local-area network (LAN) or a wide area network (WAN) such as the Internet or the World Wide Web. Users of the client nodes 10, 20 can be connected to the network 40 through a variety of connections including standard telephone lines, LAN or WAN links (e.g., T1, T3, 56kb, X.25), broadband connections (ISDN, Frame Relay, ATM), and wireless connections. The connections can be established using a variety of communication protocols (e.g., TCP/IP, IPX, SPX, NetBIOS, Ethernet, RS232, and direct asynchronous connections).

The client node 10 can be any personal computer (e.g., 286, 386, 486, Pentium, Pentium II, Macintosh computer), Windows-based terminal, Network Computer, wireless device,

information appliance, RISC Power PC, X-device, workstation, mini computer, main frame computer or other computing device that has a windows-based desktop and sufficient persistent storage for executing application programs downloaded from the application servers 30, 32, 34 across the network 40. Windows-oriented platforms supported by the client node 10 can include Windows 3.x, Windows 95, Windows 98, Windows NT 3.51, Windows NT 4.0, Windows CE, Macintosh, Java, and Unix. The client node 10 can include a display screen 12, a keyboard 14, memory 16 for storing downloaded application programs, a processor 17, and a mouse 18. The memory 16 can provide persistent or volatile storage. The processor 17 can execute the application programs locally on the client node 10 and display a resulting windows-based desktop on the display screen 12. Such local processing on the client node 10 is according to the above-described client-based computing model.

Alternatively, the client node 20 can be any terminal (windows or non-windows based), or thin-client device operating according to a server-based computing model. In a server-based computing model, the execution of application programs occurs entirely on the application servers 30, 32, 34, and the user interface, keystrokes, and mouse movements are transmitted over the network 40 to the client node 20. The user interface can be text driven (e.g., DOS) or graphically driven (e.g., Windows). Platforms that can be supported by the client node 20 include DOS and Windows CE for windows-based terminals. The client node 20 includes a display screen 22, a keyboard 24, a mouse 28, a processor (not shown), and persistent storage (not shown).

The application servers 30, 32, 34, and 36 can be any computing device that controls access to other portions of the network (e.g., workstations, printers). It is to be understood that more or fewer application servers can be connected to the network 40. The servers 30, 32, 34,

and 36 can operate according to either a client-based computing model or a server-based computing model as described above.

Each application server 30, 32, 34, and 36 hosts one or more application programs that can be accessed by the client nodes 10 and 20. Applications made available to client nodes for use are referred to as published applications. Examples of such applications include word processing programs such as MICROSOFT WORD® and spreadsheet programs such as MICROSOFT EXCEL®, both manufactured by Microsoft Corporation of Redmond, Washington, financial reporting programs, customer registration programs, programs providing technical support information, customer database applications, or application set managers.

The servers 30, 32, and 34 can belong to the same domain 38. In the network 40, a domain is a sub-network comprising a group of application servers and client nodes under control of one security database. A domain can include one or more "server farms." (A server farm is a group of servers that are linked together to act as a single server system to provide centralized administration.) Conversely, a server farm can include one or more domains. For servers of two different domains to belong to the same server farm, a trust relationship may need to exist between the domains. A trust relationship is an association between the different domains that allows a user to access the resources associated with each domain with just one log-on authentication.

In one embodiment, the application server 36 is in a different domain than the domain 38. In another embodiment, the application server 36 is in the same domain as servers 30, 32, and 34. For either embodiment, application servers 30, 32, and 34 can belong to one server farm, while the server 36 belongs to another server farm, or all of the application servers 30, 32, 34, and 36 can belong to the same server farm. When a new server is connected to the network 40, the new server joins either an existing server farm or starts a new server farm.

The network 40 can include a master server node for performing load-level balancing among the application servers 30, 32, 34, and 36. The master server node can be one of the application servers 30, 32, 34, or 36. The master server node includes a list of server addresses and load information corresponding to each of the other application servers. The master server node can direct the client node to a particular server node on which to execute an application based on the list of available servers and corresponding load levels. Alternatively, the application servers 30, 32, 34, 36 may collaborate among themselves in a peer-to-peer fashion in order to exchange administration information such as load levels, allowing any server 30, 32, 34, 36 to respond to a request made by a client node 10, 20.

Program Neighborhood

According to the principles of the invention, a user of either client node 10, 20 is able to learn of the availability of application programs hosted by the application servers 30, 32, 34, and 36 in the network 40 without requiring the user to know where to find such applications or to enter technical information necessary to link to such applications. These available application programs comprise the "program neighborhood" of the user. A system for determining a program neighborhood for a client node includes an application program (hereafter referred to as the "Program Neighborhood" application), memory for storing components of the application program, and a processor for executing the application program.

The Program Neighborhood application can be installed in memory of the client node 10 and/or on the application servers 30, 32, 34, and 36 as described below. The Program Neighborhood application is a collection of services, applications program interfaces (APIs), and user interface (UI) programs that disclose to users of the client nodes 10, 20 those application programs hosted by the application servers that each client node is authorized to use (e.g., execute).

An application server operating according to the Program Neighborhood application collects application-related information from each of the application servers in a server farm. The application-related information for each hosted application can be a variety of information including, for example, an address of the server hosting that application, the application name, the users or groups of users who are authorized to use that application, and the minimum capabilities required of the client node before establishing a connection to run the application. For example, the application may stream video data, and therefore a required minimum capability is that the client node supports video data. Other examples are that the client node can support audio data or can process encrypted data. The application-related information can be stored in a database as described later in the specification.

When a client node logs onto the network 40, the user of the client node provides user credentials. User credentials typically include the username of the client node, the password of the user, and the domain name for which the user is authorized. The user credentials can be obtained from smart cards, time-based tokens, social security numbers, user passwords, personal identification (PIN) numbers, digital certificates based on symmetric key or elliptic curve cryptography, biometric characteristics of the user, or any other means by which the identification of the user of the client node can be obtained and submitted for authentication. The server responding to the client node can authenticate the user based on the user credentials. The user credentials can be stored wherever the Program Neighborhood application is executing. When the client node 10 executes Program Neighborhood application, the user credentials can be stored at the client node 10. When an application server is executing the Program Neighborhood, the user credentials can be stored at that server.

From the user credentials and the application-related information, the server can also determine which application programs hosted by the application servers are available for use by

the user of the client node. The server transmits information representing the available application programs to the client node. This process eliminates the need for a user of the client node to set-up application connections. Also, an administrator of the server can control access to applications among the various client node users.

The user authentication performed by the server can suffice to authorize the use of each hosted application program presented to the client node, although such applications may reside at another server. Accordingly, when the client node launches (i.e., initiates execution of) one of the hosted applications, additional input of user credentials by the user may be unnecessary to authenticate use of that application. Thus, a single entry of the user credentials can serve to determine the available applications and to authorize the launching of such applications without an additional, manual log-on authentication process by the client user.

Either a client node 10, 20 or an application server can launch the Program Neighborhood application as described in connection with Figs. 2A-2C. The results are displayed on the display screen 12, 22 of the client node 10, 20. In a graphical windows-based implementation, the results can be displayed in a Program Neighborhood graphical window and each authorized application program can be represented by a graphical icon in that window.

One embodiment of the Program Neighborhood application filters out application programs that the client node 10, 20 is unauthorized to use and displays only authorized (i.e., available) programs. In other embodiments, the Program Neighborhood application can display authorized and unauthorized applications. When unauthorized applications are not filtered from the display, a notice can be provided indicating that such applications are unavailable. Alternatively, the Program Neighborhood application can report all applications hosted by the application servers 30, 32, 34, 36 to the user of a client node, without identifying which applications the client node 10, 20 is authorized or unauthorized to execute. Authorization can

be subsequently determined when the client node 10, 20 attempts to run one of those applications.

Fig. 2A shows an exemplary process by which a server launches the Program Neighborhood (PN) application and presents results of the PN application to the client node 10. The server can launch the PN application in response to a request 42 by the client node 10 for a particular application program. The request passes to the master server node, in this example server 30. The master server node 30, taking load-balancing and application availability into account, indicates (arrow 43) to the client node 10 that the sought-after application is available on server 32. The client node 10 and server 32 establish a connection (arrows 45 and 46). By this connection, the server 32 can transfer the executable code of the particular application to the client node 10, when the client node 10 and server 32 are operating according to the client-based computing model. Alternatively, the server 32 can execute the particular application and transfer the graphical user interface to the client node 10, when the client node 10 and server 32 are operating according to the server-based computing model. In addition, either the master server node 30 or the server 32 can execute the Program Neighborhood application 41 and push the results (arrows 43 or 46) back to the client node 10 so that when the client node 10 requests the Program Neighborhood application, the program neighborhood is already available at the client node 10.

Fig. 2B shows another exemplary process by which the client node 10 initiates execution the Program Neighborhood application and a server presents the results of the PN application to the client node 10. The client node 10 launches the Program Neighborhood application (e.g., by clicking on the Program Neighborhood icon 47 representing the application). The request 50 for the Program Neighborhood application is directed to the master server node, in this example server 30. The master server node 30 can execute the Program Neighborhood application, if the

application is on the master server node 30, and return the results to the client node 10. Alternatively, the master server node 30 can indicate (arrow 51) to the client node 10 that the Program Neighborhood application 41 is available on another server, in this example server 32. The client node 10 and server 32 establish a connection (arrows 53 and 54) by which the client node 10 requests execution of the Program Neighborhood application 41. The server 32 can execute the application 41 and transfer the results (i.e., the graphical user interface) to the client node 10.

Fig. 2C shows another exemplary process by which a client node 10 initiates execution of the Program Neighborhood application, in this example via the World Wide Web. A client node 20 executes a web browser application 80, such as NETSCAPE NAVIGATOR, manufactured by Netscape Communications, Inc. of Mountain View, California or MICROSOFT INTERNET EXPLORER, manufactured by Microsoft Corporation of Redmond, Washington.

The client node 20, via the web browser 80, transmits a request 82 to access a Uniform Resource Locator (URL) address corresponding to an HTML page residing on server 30. In some embodiments the first HTML page returned 84 to the client node 20 by the server 30 is an authentication page that seeks to identify the client node 20.

The authentication page allows the client node 20 to transmit user credentials, via the web browser 80, to the server 30 for authentication. Transmitted user credentials are verified either by the serve 30 or by another server in the farm. This allows a security domained to be projected onto the server 30. For example, if the server 30 runs the WINDOWS NT operating system, manufactured by Microsoft Corporation of Redmond, Washington, and the authenticating server runs the UNIX operating system, the UNIX security domain may be said to have been projected onto the server 30. User credentials may be transmitted "in the clear," or they may be encrypted. For example, user credentials may be transmitted via a Secure Socket Layer (SSL) connection,

which encrypts data using the RC3 algorithm, manufactured by RSA Data Security, Inc. of San Mateo, California.

The server 30 may verify the user credentials received from the client node 20. Alternatively, the server 30 may pass the user credentials to another server for authentication. IN this embodiment, the authenticating server may be in a different domain from the server 30. Authenticated user credentials of the client node 20 may be stored at the client node 20 in a per-session cookie or in fields that are not displayed by the web browser 80, or in any other manner common in maintenance of web pages. In some embodiments, a server farm with which the server 30 is associated may allow guest users, i.e., users that do not have assigned user credentials, to access applications hosted by servers in the farm. In these embodiments, the authentication page may provide a mechanism for allowing a client node 20 to identify that it is a guest user, such as a button or menu selection. In other of these embodiments, the server 30 may omit the authentication page entirely.

Still referring to Fig. 2C, once the client node 20 is authenticated by the server 30, the server prepares and transmits to the client node 20 an HTML page 88 that includes a Program Neighborhood window 58 in which appears graphical icons 57, 57' representing application programs to which the client node 20 has access. A user of client node 20 invokes execution of an application represented by icon 57 by clicking that icon 57.

Fig. 3A shows an exemplary process of communication among the client node 10, the master server node, in this example server 30, and the server 32. The client node 10 has an active connection 72 with the server 32. The client node 10 and server 32 can use the active connection 72 to exchange information regarding the execution of a first application program. The user credentials of the client node 10 are stored at the client node. Such storage of the user credentials can be in cache memory or persistent storage.

In this embodiment, the Program Neighborhood application runs on the client node 10. The client node display has a Program Neighborhood window 58 in which appears a graphical icon 57 representing a second application program. A user of the client node 10 can launch the second application program by double-clicking the icon 57 with the mouse. The request passes to the master server node 30 via a connection 59. The master server node 30 indicates to the client node 10 via the connection 59 that the sought-after application is available on server 32. The client node 10 signals the server 32 to establish a second connection 70. The server 32 requests the user credentials from the client node 10 to authenticate access to the second application program. Upon a successful authentication, the client node 10 and server 32 establish the second connection 70 and exchange information regarding the execution of the second application program. Accordingly, the client node 10 and the server 32 communicate with each other over multiple connections.

Fig. 3B shows an exemplary process of communication among the client node 20, the master server node, in this example server 30, and servers 32, 34, and 36. The client node 20 has an active connection 73 with the server 32. The client node 20 and server 32 can use the active connection 73 to exchange information regarding the execution of a first application program. The user credentials of the client node 20 are stored at the server 32 in cache memory or in persistent storage.

In this embodiment, the Program Neighborhood application runs on the server 32. The server 32 includes software providing a server-based client engine 62, enabling the server 32 to operate in the capacity of the client node 20. The client node 20 display has a Program Neighborhood window 58 in which appears graphical icons 57, 57' representing a second application program and a third application program, respectively. A user of the client node 20 can launch the second application program by double-clicking the icon 57. The request to launch

the second application program passes to the server 32 via active connection 73, and the server 32 forwards the request to the master server node 30 (arrow 65).

The master server node 30 indicates (arrow 65) to the server 32 that the sought-after application is available on server 34. The server 32 contacts the server 34 to establish a connection 66. To authenticate access to the application, the server 34 obtains the user credentials of the client node 20 from the server 32. The server 32 and server 34 establish the connection (arrow 66) by which the server 32 requests execution of the second application and the server 34 returns the graphical user interface results to the server 32. The server 32 forwards the graphical user interface results to the client node 20, where the results are displayed. Accordingly, the information exchanged between the client node 20 and the server 34 "passes through" the server 32.

Similarly, the client node 20 can launch the third application program by double-clicking the icon 57'. The request to launch the third application program passes to the server 32. The server 32 forwards the request to the master server node 30, which considers load-balancing and application program availability to determine which server can handle the request. In this example, the master server node indicates that server 36 can run the third application program.

The server 32 and server 36 establish a connection (arrow 74) by which the server 32 requests execution of the third application program, and the server 36 returns the graphical user interface results to the server 32. To permit execution of the third application program, the server 36 can authenticate the user credentials of the client node 20, which are obtained from the server 32. The server 32 forwards the graphical user interface results to the client node 20 where the results are displayed. Accordingly, the results of executing the third application program pass between the client node 20 and the server 36 through the server 32.

From this illustration it should be understood that client node 20 can run multiple application programs through one connection with the server 32, while the server 32 maintains multiple connections (in this example, one connection with server 34 and a second connection with server 36). Also, the server 32 merges the information received from the server 34 with the information received from the server 36 into one data stream for transmission to the client node 20.

Fig. 3C shows an exemplary process of communication among the client node 20, the master server node, in this example server 30, and servers 32 and 34. The client node 20 has an active connection 76 with the server 32. The client node 20 and server 32 can use the active connection 76 to exchange information regarding the execution of a first application program. The client node 20 can store the user credentials in cache memory or in persistent storage.

In this embodiment, the Program Neighborhood application runs on the server 32. The client node 20 display has a Program Neighborhood window 58 in which appears a graphical icon 57 representing a second application program. A user of the client node 20 can launch the second application program by double-clicking the icon 57. The request to launch the second application program passes to the server 32. The server 32 responds (i.e., "calls back") to the client node 20 by returning application-related information such as the name of the application and capabilities needed by the client node 20 for the second application to run.

With the information provided by the server 32, the client node 20 then communicates with the master server node 30 via connection 77 to determine the server for executing the second application program. In this example, that server is server 34. The client node 20 then establishes a connection 78 to the server 34. Server 34 requests the user credentials from the client node 20 to authenticate the user of the client node 20. The second application program executes on the server 34, and the server 34 returns the graphical user interface to the client node

20 via the established connection 78. Accordingly, the client node 20 can have multiple active connections between the multiple servers.

Fig. 3D shows an exemplary process of communicating between the client node 20, a server 30 that in this example acts as a web server, and server 32. The client node 20 authenticates itself to the server 30 as described above in connection with Fig. 2C. In one embodiment, the sever 30 accesses an output display template 90, such as an SGML, HTML or XML file, to use as a base for constructing the Program Neighborhood window to transmit to the client node 20. The template may be stored in volatile or persistent memory associated with the server 30 or it may be stored in mass memory 92, such as a disk drive or optical device, as shown in Fig. 3D.

In this embodiment, the template 90 is a standard SGML, HTML, or XML document containing Program Neighborhood-specific tags that are replaced with dynamic information. The tags indicate to the server 30 where in the output display to insert information corresponding to available applications, such as icon images. In one particular embodiment, the Program Neighborhood-specific tags are embedded within comments inside the file, allowing the file to remain compatible with standard interpreters. In another embodiment, the Program Neighborhood-specific tags are extensions of the markup language used as the base for the template.

Examples of HTML tags that may be used in a template in accordance with the present invention are set forth below in Table 1:

TABLE 1

Tag	Description
ControlField *field value*	This tag is used to set the value of data that either persists between Program Neighborhood web pages, are set by the user, or are used to help in cross page navigation, such as user name, domain, password, template, and application.
DrawProgramNeighborhood	His tag is used to draw a Program Neighborhood display at this location in an output display.
AppName	This tag is replaced by the name of the published application in the current context.
WindowType	This tag is replaced by the window type of the published application in the current context.
WindowHeight	This tag is replaced by the window height of the published application in the current context.
WindowWidth	This tag is replaced by the window width of the published application in the current context.
WindowScale	This tag is replaced by the window scale of the published application in the current context.
NumWindowColors	This tag is replaced by the color depth of the published application in the current context.
SoundType	This tag is replaced by the sound setting of the published application in the current context.
VideoType	This tag is replaced by the video setting of the published application in the current context.
EncryptionLevel	This tag is replaced by the encryption level of the published application in the current context.
Icon	This tag is replaced by the icon of the published application in the current context.

Other tags can be provided to set control fields and to provide conditional processing relating to the Program Neighborhood application.

In one embodiment, the template is constructed dynamically using, for example, COLD FUSION, manufactured by Allaire Corp. of Cambridge, Massachusetts or ACTIVE SERVER PAGES manufactured by Microsoft Corporation of Redmond, Washington. Alternatively, the template may be static. The Program Neighborhood application parses the template, replacing Program Neighborhood-specific tags as noted above. Tags that are not Program Neighborhood-specific are left in the file to be parsed by the browser program 80 executing on the client 20.

In one embodiment, a template parser object is provided that accepts an HTML template as input, interprets Program Neighborhood-specific tags present in the template, and output the original template with all Program Neighborhood tags replaced with appropriate text. The template parser object can be passed a cookie, a URL query string, or a control field from a web server interface to provide the information with which Program Neighborhood-specific tags should be replaced.

In another embodiment, the Program Neighborhood application allows scripts to accept to information via an application programming interface. Scripts may be written in, for example, VBScript or Jscript. In this embodiment, the scripting language is used to dynamically generate an output display using information returned by the application in response to queries posed by the script. Once the output display is generated, it is transmitted to client node 20 for display by the browser program 80.

A user of the client node 20 can launch an application by double-clicking with a mouse an icon 57, 57' displayed in the Program Neighborhood web page. In some embodiments, each icon 57, 57' is an encoded URL that specifies: the location of the application (i.e., on which servers it is hosted or, alternatively, the address of a master server); a launch command associated with the application; and a template identifying how the output of the application should be displayed (i.e., in a window "embedded" in the browser or in a separate window). In

some embodiments, the URL includes a file, or a reference to a file, that contains the information necessary for the client to create a connection to the server hosting the application. This file may be crated by the Program Neighborhood application dynamically. The client node 20 establishes a connection (arrow 94) with the server identified as hosting the requested application (in this example, server 32) and exchanges information regarding execution of the desired application. In some embodiments, the connection 94 is made using the Independent Computing Architecture (ICA) protocol, manufactured by Citrix Systems, Inc. of Fort Lauderdale, Florida. Thus, the client node 20 may display application output in a window separate from the web browser 60, or it may "embed" application output within the web browser.

Fig. 4 illustrates an exemplary arrangement of program components for a client-based implementation of the Program Neighborhood application. A client-based implementation of Program Neighborhood application can be used in a network using either the server-based computing model in which the servers execute the Program Neighborhood application or the client-based computing model in which the client node 10 executes the Program Neighborhood application locally. The Program Neighborhood application includes a Program Neighborhood Service (PNSVC) component 44, an Application Database component 48, a Program Neighborhood Application Program Interface (PNAPI) component 52, a Program Neighborhood User Interface component 56, and a local cache 60.

The application server 30, for example, includes the service component (PNSVC) 44 and the application database 48. The client node 10, which is a representative example of a client node that can support a client-based implementation of the Program Neighborhood application, includes the application program interface PNAPI 52, the user interface user interface component 56, and the local cache 60 components. The PNAPI 52 communicates with the user interface

component 56 and the local cache 60. The PNSVC 44 communicates with the application database 48 and with the PNAPI 52 on the client node 10 via communications link 62.

The communications link 62 can be established by, for example, using the ICA protocol. ICA is a general-purpose presentation services protocol designed to run over industry standard network protocols, such as TCP/IP, IPX/SPX, NetBEUI, using industry-standard transport protocols, such as ISDN, frame relay, and asynchronous transfer mode (ATM). The ICA protocol provides for virtual channels, which are session-oriented transmission connections that can be used by application-layer code to issue commands for exchanging data. The virtual channel commands are designed to be closely integrated with the functions of client nodes. One type of virtual channel connection supported by the ICA protocol is a Program Neighborhood virtual channel.

The Program Neighborhood virtual channel protocol can include four groups of commands:

(1) initialization-related commands;

(2) single authentication related commands that can be supported by each client node wanting a copy of the user credentials;

(3) application data related commands for implementing the Program Neighborhood user interface; and

(4) application launch callback-related commands for running the user interface on an application server.

Application Database

The application database 48 is a cache of the authorized user and group information for all the public (i.e., published) applications in a server farm or in a group of trusted domains. Each server in a server farm can maintain its own application-related information in persistent

storage and build up the database 48 in volatile storage. In another embodiment, all collected application-related information in the database 48 can be stored in persistent storage and made accessible to each other server in the server farm. The database 48 can be implemented in a proprietary format (e.g., as a linked list in memory) or using Novell's Directory Services (NDS) or any directory service adhering to the X.500 standard defined by the International Telecommunication Union (ITU) for distributed electronic directories.

The application database 48 includes a list of application servers. Each server in the list has an associated set of applications. Associated with each application is application-related information that can include the application name, a list of servers, and client users that are authorized to use that application. An overly-simplified example of the application-related information maintained in the database is illustrated by the following Table 1. Users A and B are users of the client nodes 10, 20, "n/a" indicates that the application is hosted, but is not available to client node users, and "-" indicates that the application is not hosted.

TABLE 2

	Applications			
Server Name	SpreadSheet	Customer Database	Word Processor	Calculator
Server 30	User A	User B	n/a	-
Server 32	User B	n/a	User A	-
Server 34	-	-	-	User A User B

Table 1 shows a list of servers 30, 32, 34, the applications hosted by the servers, (SpreadSheet, Customer Database, Word Processor, and Calculator), and those users who are authorized to use the applications. For example, server 30 hosts the SpreadSheet program, the

Customer Database and the Word Processor. User A is authorized to use the SpreadSheet, User B is authorized to use the Customer Database, and no users are authorized to use the Word Processor. It is to be understood that other techniques can be used to indicate who is authorized to use a particular application. For example, the user information stored in the database can be used to indicate those users who are unauthorized to use a particular application rather than those who are authorized.

To obtain the information that is stored in the database 48, the server 30 obtains the application-related information from each other server in the server farm regarding the applications on those servers, including control information that indicates which client users and servers are permitted to access each particular application. The application-related information maintained in the database may or may not persist across re-boots of the server 30.

The application database 48 can be a central database that is stored at the application servers 30 and is accessible to all of the servers in the server farm. Accordingly, the application-related information can be available for use by other servers such as those servers that perform published application authentication during session log-on and application launching. In another embodiment, the application database 48 can be maintained at each of the application servers based upon the information that each server obtains from communications with each other server in the server farm.

Program Neighborhood Service Program (PNSVC)

Each server 30, 32, 34 and 36 having the Program Neighborhood application installed thereon executes the PNSVC software 44. The PNSVC software 44, operating on each server 30, 32, 34 and 36 establishes a communication link (e.g., a named pipe) with each other server. The servers 30, 32, 34 and 36 exchange the application-related information on the named pipes. In another embodiment, the PNSVC software 44 collects the application-related information

from the other servers in the server farm through remote registry calls (e.g., the service component 44 transmits a datagram to other servers in the plurality requesting the application-related information corresponding to the application programs hosted by those servers). The PNSVC 44 software also maintains the relationships of groups and users to published applications in the application database 48 and accesses the information when authenticating a client user. An administrator of the server 30 can use a user interface to configure the PNSVC 44.

Other functions of the PNSVC software 44 include implementing the services and functions requested by the PNAPI 52 and communicating with the PNAPI 52 on the client node 10 using a Program Neighborhood virtual device driver (VDPN). The VDPN operates according to the Program Neighborhood virtual channel protocol described above for establishing and maintaining an ICA connection.

Program Neighborhood Application Program Interface (PNAPI)

The PNAPI 52 is a set of software functions or services that are used by the Program Neighborhood application to perform various operations (e.g., open windows on a display screen, open files, and display message boxes). The PNAPI 52 provides a generic mechanism for launching application objects (e.g., icons) produced by running the Program Neighborhood application and application objects in a legacy (i.e., predecessor or existing for some time) client user interface. When the client node 10 launches an available application, the launch mechanism can launch the application on the server 30, if necessary (e.g., when the client node 10 does not have the resources to locally execute the application).

The PNAPI 52 provides all published application information to the user interface component 56 for display on the screen 12 of the client node 10. The PNAPI 52 also manages server farm log-ons in a local database of logon credentials (e.g., passwords) for users of the

client node 10 to support the single authentication feature. Credentials may or may not be persistent across bootings (power-off and on cycles) of the client node 10.

The PNAPI 52 provides automatic and manual management for Program Neighborhood application objects stored in the local cache 60. The local cache 60 can either be refreshed manually by the user of the client node 10, or at a user-definable refresh rate, or by the server at any time during a connection. In a Windows implementation, the PNAPI 52 can build remote application file associations and manage the "Start" menu and desktop icons for application object shortcuts.

Program Neighborhood User Interface

The user interface module 56 interfaces the PNAPI 52 and can be a functional superset of an existing client-user interface (e.g., Remote Application Manager). The user interface module 56 accesses the information stored in the local cache 60 through the PNAPI 52 and visually presents that information to the user on the display screen 12 of the client node 10. The displayed information is a mixture of information generated by a user of the client node 10 and information obtained by the Program Neighborhood application. The user interface module 56 can also show the user all applications that the user is currently running and all active and disconnected sessions.

In a windows-based embodiment, the user interface module 56 can present a variety of graphical components, such as windows and pull-down menus, to be displayed on the display screen 12. A display of a combination of such graphical user interface components is generally referred to as a "desktop." A desktop produced by the user interface module 56 can include a Program Neighborhood window displaying the neighborhood of application programs available to the user of the client node 10 for use. These application programs are a filtered combination of the published applications hosted by a server farm on the network. The user interface module

56 can generate a Program Neighborhood window for each server farm or merge the applications from different server farms under a single Program Neighborhood window.

At a top level, the Program Neighborhood window includes a folder for each server farm. Clicking on one of the folders with the mouse 18 produces a window containing a representation (e.g., an icon) of each hosted application available to the user, e.g., see Figs. 6A and 6B. The Program Neighborhood window becomes the focal point for launching published applications, and the user interface module 56 can be used to launch applications through the PNAPI 52. For example, the user of the client node 10 can use the mouse 18 to select one of the displayed icons and launch the associated application.

A feature of a client-based implementation is that the user can browse the objects displayed in the Program Neighborhood window although the client node is offline, that is, the ICA connection 62 is inactive. Also, a user of the client node 10 can drag application objects and folders out of the Program Neighborhood window and into other graphical components (e.g., other windows, folders, etc.) of the desktop.

Fig. 5 shows an exemplary arrangement of program components for a server-based implementation of the Program Neighborhood application. The components include a Service (PNSVC) component 44', an Application Database component 48', an Application Program Interface (PNAPI) component 52', a User Interface component 56' and a local cache 60'. Each software component 44', 48', 52', 56', and 60' is installed on the application server 30'. The software components for the server-based implementation correspond to the software components for the client-based implementation of Fig. 4. The functionality of each server-based software component is similar to the client-based counterpart, with differences or added capabilities described below. The PNSVC 44' communicates with the application database 48'

and with the PNAPI 52' using local procedure calls. The PNAPI 52' also communicates with the user interface module 56' and the local cache 60'.

Similar to that described in Fig. 4 for the client node 10, the client node 20 logs on to the network 40, the server 30' develops and maintains a database containing the application related information collected from the other servers 32, 34 in the server farm, and a communication link is established between the server 30' and the client node 20. The application server 30' is in communication with the client node 20 via an ICA channel connection 62'. The channel connection 62' can be established by an ICA virtual channel protocol (e.g., Thinwire). The Thinwire protocol can be used to transmit presentation commands from Windows-based applications running on the application server 30' to the client node 20. To a user of the client node 20, the applications appear to be running on the client node 20. The client node 20 can include a Remote Application Manager application program 64 that communicates with the application server 30' via the ICA channel connection 62'.

To run the Program Neighborhood application in a server-based implementation, the user of the client node 20 connects to an initial desktop (at the server 30') and launches the Program Neighborhood application from within that desktop environment. The connection to the initial desktop can occur automatically, e.g., via a logon script of the client node 20, via an entry in the StartUp group in Windows 95, or by another centrally managed server specific mechanism. All remote application management and launching is accomplished through this initial desktop.

Similar to that described in Fig. 4 for the server 30, the server 30' uses the user credentials to determine those application programs that are authorized for use by the user of the client node 20. A Program Neighborhood graphical window is returned to the client node 20 and displayed on the client screen 22. This window can contain icons representing the available and,

possibly, the unavailable application programs that are in the program neighborhood of the client node 20.

The user of the client node 20 can select and launch one of the application programs displayed in the Program Neighborhood window. When launching an application, the Program Neighborhood application can execute the application on the same server 30', where applicable, taking into account load balancing requirements among servers and the availability of the application on that server 30'. The PNAPI 52' can include a launch mechanism for launching a remote application locally on the server 30' when the server 30' is nominated to launch the application. When a different server is needed to run the application, the Program Neighborhood application can launch the application via the server 30' (i.e., server-based client) using the windows to present the application on the desktop of the client node 20 as described above in Fig. 3B.

In one embodiment, the web-based Program Neighborhood application includes a group of objects that manage various aspects of the application. In one embodiment, the application includes three primary object classes that "plus in" to a web server: a gateway object class; a credentials object class; and an applications object class. In some specific embodiments, the object classes are provided as Java beans. The three primary object classes facilitate: validation of user credentials into a server farm; generation of lists of published applications that a specified user may access; provision of detailed information about a specific published application; and conversion of published application information into an ICA-compatible format.

When provided as Java beans, the objects can be accessed in a number of different ways. For example, they may be compiled as COM objects and made available to the web server as ActiveX components. In another embodiment, the Java beans can be used in their native form, such as when the server uses Java Server Pages technology. In yet another embodiment, the Java

beans can be instantiated and used directly in a Java servlet. In still another embodiment, the server 30 can instantiate the Java beans as COM objects directly.

A credentials object class manages information necessary to authenticate a user into a target server farm. A credentials object passes stored user credentials to other Program Neighborhood objects. In some embodiments, the credentials object is an abstract class that cannot be instantiated and represents a user's credentials. Various class extensions may be provided to allow different authentication mechanisms to be used, including biometrics, smart cards, token-based authentication mechanisms such as challenge-response and time-based password generation, or others. For example, a "clear text credentials" extension may be provided that stores user's name domain, and password in plain text.

A gateway object class handles communications with a target server farm. In one embodiment, the gateway object class is provided as an abstract Java class that cannot be instantiated. A particular gateway object may retrieve application information by communicating with a server farm using a particular protocol, reading cached application information, a combination of these two methods, or other various methods.

As noted above, the gateway object class caches information to minimize communication with a target server farm. Extensions to the gateway object may be provided to communicate with the server farm over specific protocols, such as HTTP. IN one embodiment, an extension class is provided that allows the gateway object to communicate with the server farm via WINDOWS NT named pipes. The gateway object may provide an application programming interface hook that allows other Program Neighborhood objects to query the object to application information.

An application object class contains information about published applications and returns information about applications hosted by the server farm in order to create the Program

Neighborhood web page. The applications object class creates objects representing applications by retrieving information relating to the applications, either from an object created by the gateway object or directly from the servers in the server farm. An application object acts as a container for certain properties of the application, some settable and some not settable, such as: the name of the application (not settable); the percentage of the client's desktop that the client window should occupy (settable); the width of the client window, in pixels, for this application (settable); the height off the client window, in pixels, for this application (settable); the number of colors to use when connecting to the application (settable); the severity of audio bandwidth restriction (settable); the level of encryption to use when connecting to the application (settable); the level of video to use when connecting to this application (settable); whether the application should be place don a client's start menu (settable); whether the application should be placed on the client's desktop (settable); the identity of the Program Neighborhood folder to which the application belongs (settable); the description of the application (settable); the source of the graphics icon file for the application (settable); the type of window that should be used when connecting to the application (not settable); and whether to override default parameters for the object.

The applications base class contains information about published applications and is used by the gateway base class to return information about applications hosted by the server farm in order to create the Program Neighborhood web page.

Fig. 6A is a screenshot of an exemplary Program Neighborhood window 120 that can be displayed on the screen 18, 22 of either client node 10, 20 after the Program Neighborhood application has executed. The window 120 includes graphical icons 122. Each icon 122 represents an application program that is hosted by one of the servers 30, 32, 34 and 36 on the network 40. Each represented application is available to the user of the client node for execution.

The user can select and launch one of the applications using the mouse 18, 28 or keyboard 14, 24.

Fig. 6B is a screenshot of another exemplary Program Neighborhood window 124 that can be displayed on the screen 18, 22 of either client node 10, 20 after the Program Neighborhood application has executed. The window 124 includes graphical icons 126, 128. Each icon 126, 128 represents an application program that is hosted by one of the servers 30, 32, 34 and 36 on the network 40. Each application program represented by one of the icons 126 is available to the user of the client node 10, 20 for execution. The user can select and launch one of the applications using the mouse 18, 28 or keyboard 14, 24. For web-based program neighborhood environments, the screenshots of Figs. 6A and 6B are similar, except that icons 122, 166, 128 are displayed within a browser window.

Each application program represented by one of the icons 128 is unavailable to the user of the client node 10, 20, although such applications are present in the server farm. The unavailability of these application programs can be noted on the display screen (e.g., "X"s can be drawn through the icons 128). An attempt to launch such an application program can trigger a message indicating that the user is not authorized to use the application.

Fig. 7 shows an exemplary process by which a user of either client node 10, 20 can be informed about the availability of applications hosted by application servers 30, 32, 34 and 36 on the network 40. In step 80, the client node 10, 20 requests log-on service from one of the servers, e.g., server 32. The server 32 requires valid user credentials to establish the connection. The server 32 receives the user credentials from the client node 10 (step 82), and authenticates the user for log-on (step 84). A desktop is displayed at the client node 10, 20 (step 85). The desktop can include a graphical icon representing the Program Neighborhood application program.

In step 86, the application server 32 establishes a connection with each other servers 30, 34 and 36 to exchange application-related information, as described above, corresponding to application programs hosted on those servers (step 88). In step 90, the application server 32 develops and maintains a database of the collected application-related information. Each other server 30, 34, 36 in the server farm can develop a database equivalent to the database of the server 32 and in similar manner as the server 32. In another embodiment, the database of the server 32 can be a centralized database that is accessible to each other application server 30, 34, 36 in the server farm. The collecting of application-related information can occur independently or be triggered by the request of the client node 10, 20 to log-on to the server farm 38.

In step 92, the client node 10, 20 can request execution of an application program from the desktop display. The master server node can process the request and, using a load-balancing evaluation and application availability as described above, determine the application server to provide the service to the client node 10, 20 (step 94). For example, the application server 32 can be selected to service the request with the client node 10, 20. In step 96, the client node 10, 20 establishes a communications link with the server 32. The server 32 and the client node 10, 20 can communicate according to the ICA protocol appropriate for that client node as described above.

Also in response to this request to run the application program, the master server node 30 or the server 32 can run the Program Neighborhood application (step 93) and push the results to the client node 10, 20, although the client node 10, 20 may not have requested the PN application program. When executing the PN application program, the master server node 30 or server 32 filters the application-related information in the database using the user credentials (step 100). The result of filtering the database determines those application programs that are authorized for use by the user of the client node 10, 20. The authorized application programs are in the program

neighborhood of the client node 10, 20. This program neighborhood of available application information is pushed to the client node 10, 20 (step 102) and displayed on the client screen 12, 22 in a Program Neighborhood graphical window (step 104).

In other embodiments, the Program Neighborhood window may include applications that are in the server farm but unavailable for use to the client node 10, 20. In a Windows-based implementation, the available (and unavailable) application programs can be represented by icons. The user of the client node 10, 20 can select and launch one of the application programs displayed in the Program Neighborhood window.

The present invention may be provided as one or more computer-readable programs embodied on or in one or more articles of manufacture. The article of manufacture may be a floppy disk, a hard disk, a CD ROM, a flash memory card, a PROM, a RAM, a ROM, or a magnetic tape. In general, the computer-readable programs may be implemented in any programming language. Some examples of languages that can be used include C, C++, or JAVA. The software programs may be stored on or in one or more articles of manufacture as object code.

While the invention has been shown and described with reference to specific preferred embodiments, it should be understood by those skilled in the art that various changes in form and detail may be made therein without departing from the spirit and scope of the invention as defined by the following claims.

What is claimed is:

1. In a network including a client system and a plurality of servers including a host server, the plurality of servers hosting application programs, a method for presenting to the client system application programs that are available for use, the method comprising the steps of:

(a) receiving at the host server application-related information corresponding to application programs hosted by the plurality of servers;

(b) receiving user credentials from the client system;

(c) determining for each application program hosted by the plurality of servers whether that hosted application program is available to the client system based on the user credentials and the application-related information; and

(d) transmitting information from the host server to the client system indicating to the client system each hosted application program that is available to the client system for execution.

2. The method of claim 1 further comprising the step of establishing a connection between the client system and the host server using an Independent Computing Architecture (ICA) protocol.

3. The method of claim 2 wherein the ICA protocol establishes a virtual channel to transmit the information to the client system indicating each hosted application program that is available to the client system.

4. The method of claim 1 further comprising the steps of:

establishing a connection between the client system and the host server;

receiving a request to execute one of the available hosted application programs; and

establishing in response to the request a second connection between the client system and the host server to exchange information associated with an execution of the requested application program.

5. The method of claim 1 further comprising the steps of:

establishing a first connection between the client system and the host server;

receiving a request to execute one of the available hosted application programs;

establishing in response to the request a second connection between the host server and a second server hosting the requested application; and

exchanging information associated with an execution of the requested application program between the second server and the client system via the host server.

6. The method claim 5 further comprising the steps of:

receiving a second request to execute a second one of the available hosted application programs;

establishing in response to the second request a third connection between the host server and a third server hosting the second requested application; and

exchanging information associated with an execution of the second requested application program between the third server and the client system via the host server.

7. The method of claim 6 further comprising the step of merging information received from the second and third servers for transmission to the client system via the first connection.

8. The method claim 1 further comprising the steps of:

establishing a first connection between the client system and the host server;

receiving a request to execute one of the available hosted application programs;

providing application information to the client system in response to the request; and

establishing, in response to the provided application information, a second connection between the client system and a second server hosting the requested application.

9. The method of claim 1 wherein the transmitted information displays at the client system icons representing the available application programs in a graphical user interface window.

10. The method of claim 1 further comprising the step of communicating, by the host server, with each of the other servers in the plurality to determine application programs hosted by those other servers.

11. The method of claim 1 further comprising the step of establishing a communications link between the host server and the client system, wherein the communications link includes at least one virtual channel.

12. The method of claim 1 further comprising the steps of:

receiving user credentials from the client system;

authenticating at the host server the user of the client system based on the received user credentials; and

executing a selected one of the available application programs hosted by one of the plurality of servers other than the host server without requiring further input of user credentials by the user of the client system.

13. The method of claim 1 wherein the host server transmits the available application information in response to a request for the available application information by the client system.

14. The method of claim 1 further comprising the step of maintaining a database at the host server for storing the application-related information, wherein the database is accessible to each of the other servers of the plurality.

15. The method of claim 1 further comprising the step of maintaining a database at each of the plurality of servers for storing the application-related information.

16. The method of claim 1 wherein the transmitted information further discloses to the user of the client system each hosted application program that the user of the client system is unauthorized to use.

17. In a network including a server hosting at least one application, a method for displaying each application available to a client system, the method comprising the steps of:

(a) receiving user credentials from the client system;

(b) maintaining application-related information corresponding to each application program hosted by the server;

(c) determining whether each application program hosted by the server is available for use by a user of the client system based on the user credentials and the application-related information; and

(d) transmitting information from the server to the client system responsive to the application-related information, the transmitted information disclosing each application program that is determined to be available for use by the user of the client system.

18. In a network including a client system and a plurality of servers hosting applications, a server comprising:

a service module collecting application-related information corresponding to application programs hosted by the plurality of servers;

a receiver receiving user credentials from the client system;

a database storing the collected application-related information wherein the service module determines for each application program hosted by the plurality of servers whether that hosted application program is available to the client system for execution based on the user credentials and the application-related information stored in the database; and

a transmitter transmitting information to the client system indicating to the client system each hosted application program that is available to the client system for execution.

19. The server of claim 18 wherein said service module transmits a datagram to other servers in the plurality to collect the application-related information corresponding to the application programs hosted by those servers.

20. The server of claim 18 wherein the database includes client information.

21. The server of claim 18 wherein said transmitter transmits available application information using a virtual channel communications protocol.

22. The method of claim 1 wherein step (d) comprises:

(d-a) creating an output display indicating each hosted application program that is available to the client system for execution; and

(d-b) transmitting the created output display to the client system.

23. The method of claim 22 wherein step (a) further comprises receiving by a web server application-related information corresponding to application programs hosted by at least one of a plurality of servers.

24. The method of claim 23 further comprising the steps of:

(e) parsing an SGML document;

(f) retrieving, responsive to the parsed SGML document, application-related information corresponding to application programs hosted by at least one of the plurality of servers.

25. The method of claim 22 wherein step (b) comprises receiving by the host system biometric user credentials from a client system.

26. The method of claim 22 wherein step (b) comprises receiving by the host system encrypted user credentials from a client system.

27. The method of claim 22 wherein step (d-a) further comprises:

(a) receiving by the host system user credentials from a client system;

(b) authenticating at the host server the user of the client system based on the received user credentials; and

(c) executing a selected one of the available application programs hosted by one of the plurality of servers without requiring further receipt of user credentials from the client system.

28. The method of claim 22 wherein step (d-a) comprises creating an SGML output display indicating each hosted application program that is available to the client system for execution.

29. The method of claim 22 wherein step (d-a) comprises creating an HTML output display indicating each hosted application program that is available to the client system for execution.

30. The method of claim 22 wherein step (d-a) comprises creating an output display representing the available application programs as icons in a graphical user interface window.

31. The method of claim 22 wherein step (d-b) comprises transmitting the created output display to the client system using HTTP

32. The method of claim 22 further comprising the step of:

(e) receiving a request to execute one of the available hosted application programs; and

(f) executing the requested application program.

33. The method of claim 32 wherein step (f) further comprises executing the requested application in a window contained in the output display page.

34. The method of claim 32 further comprising the step of establishing a connection between the client system and the server hosting the requested application.

35. The method of claim 22 further comprising the step of communicating, by the host server, with the plurality of servers in order to determine application programs hosted by the plurality of servers.

36. The method of claim 17 wherein step (d) comprises:

(d-a) creating, responsive to the determination, an output display indicating each hosted application program that is available to the client system for execution; and

(d-b) transmitting the created output display to the client system.

37. The network of claim 18 wherein the server further comprises:

an output display creation engine creating output displays indicating each hosted program available to the client system for execution; and

a transmitter transmitting created output displays to the client system.

38. The server of claim 37 wherein said service module transmits a datagram to the plurality of servers to collect the application-related information corresponding to the application programs hosted by those servers.

39. The server of claim 37 wherein said output display creation engine parses SGML document templates.

40. The server of claim 39 wherein said transmitter transmits available application information using HTTP.

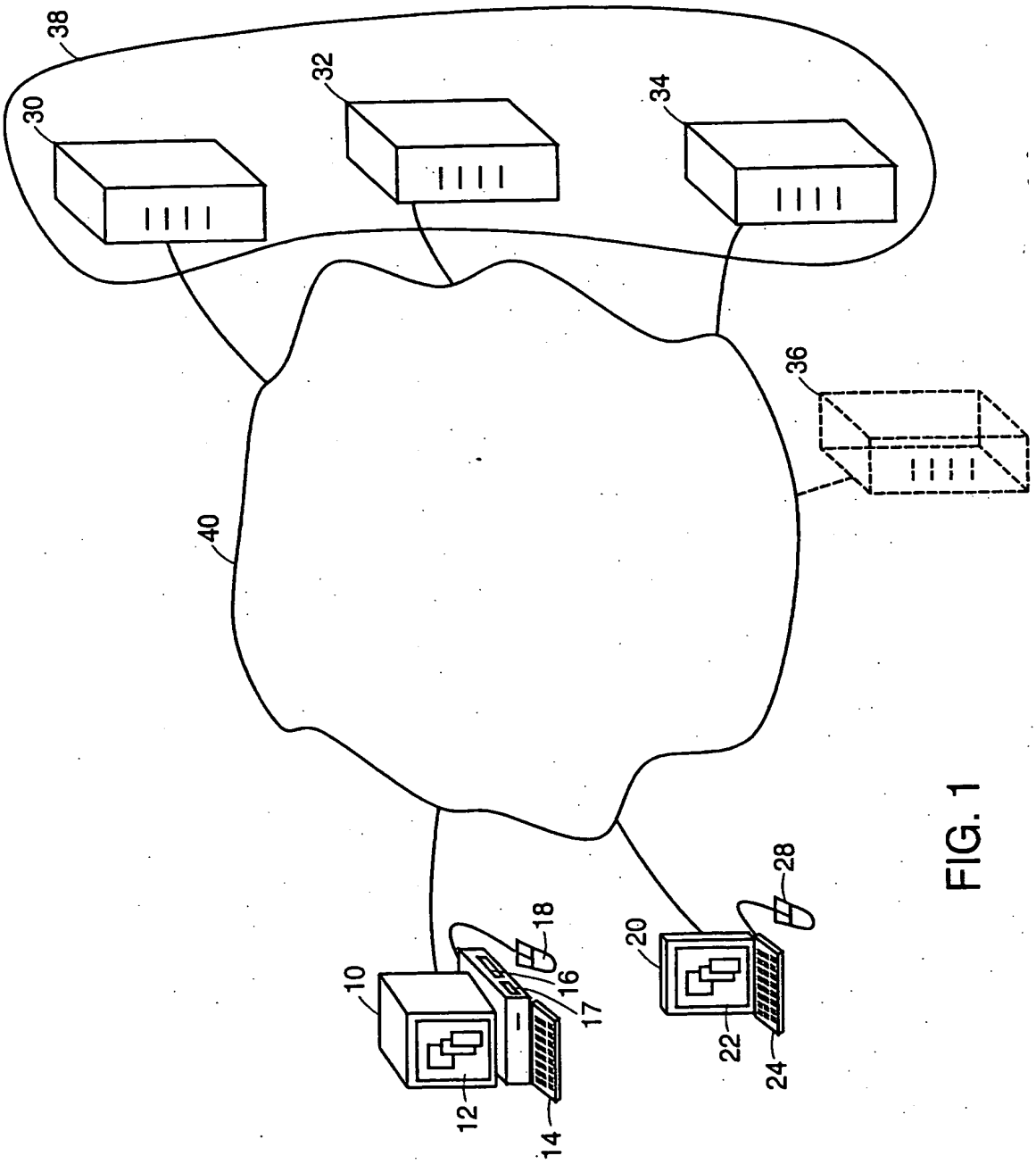


FIG. 1

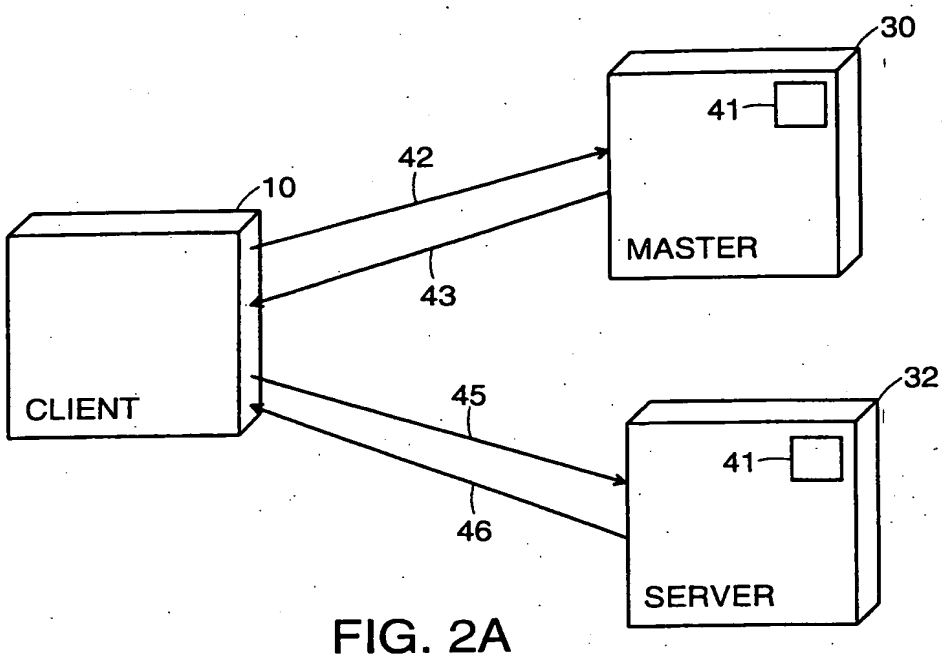

30
41
42
10
MASTER
43
CLIENT
45
32
41
46
SERVER

FIG. 2A

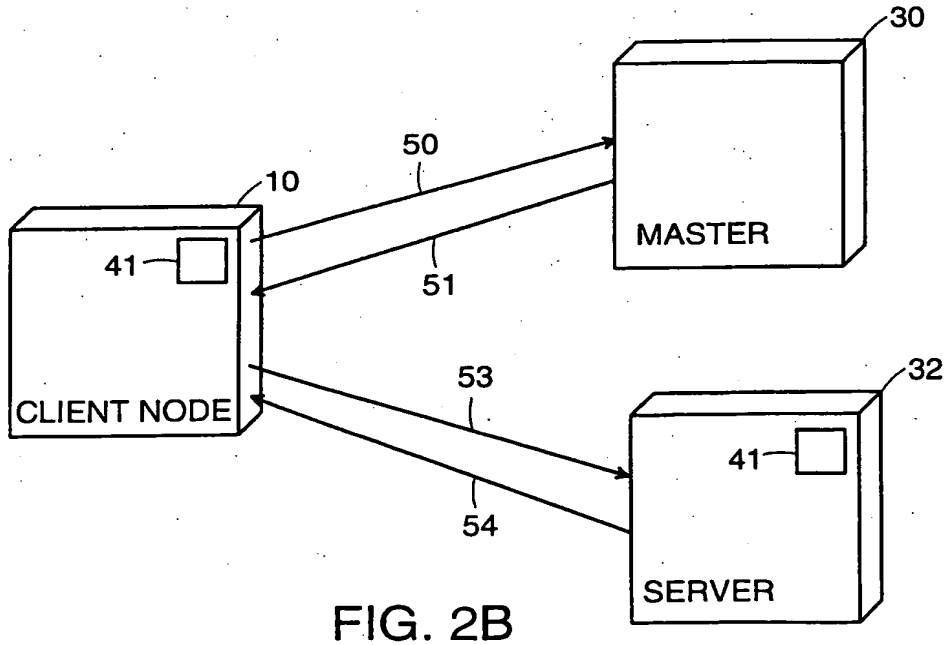

30
50
10
MASTER
41
51
CLIENT NODE
53
32
41
54
SERVER

FIG. 2B

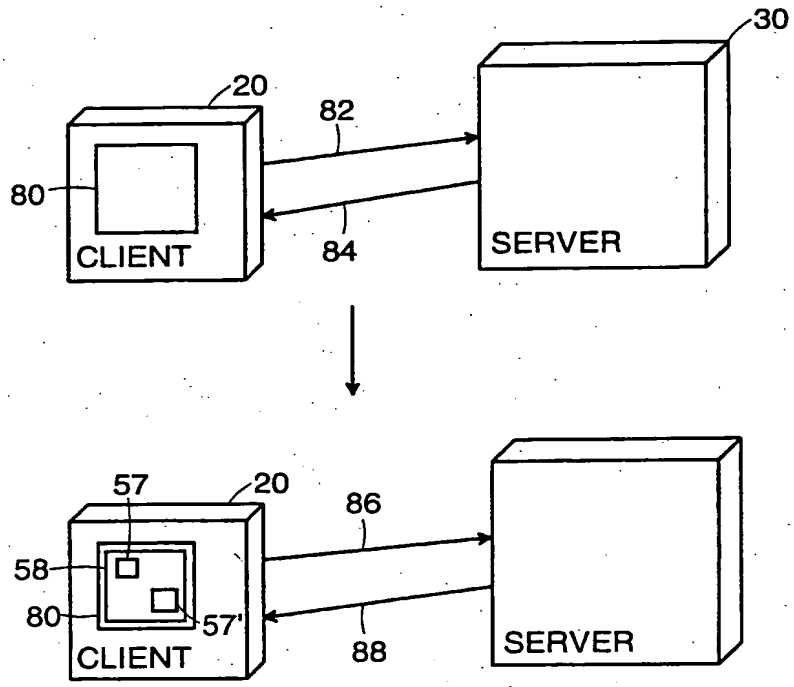


FIG. 2C

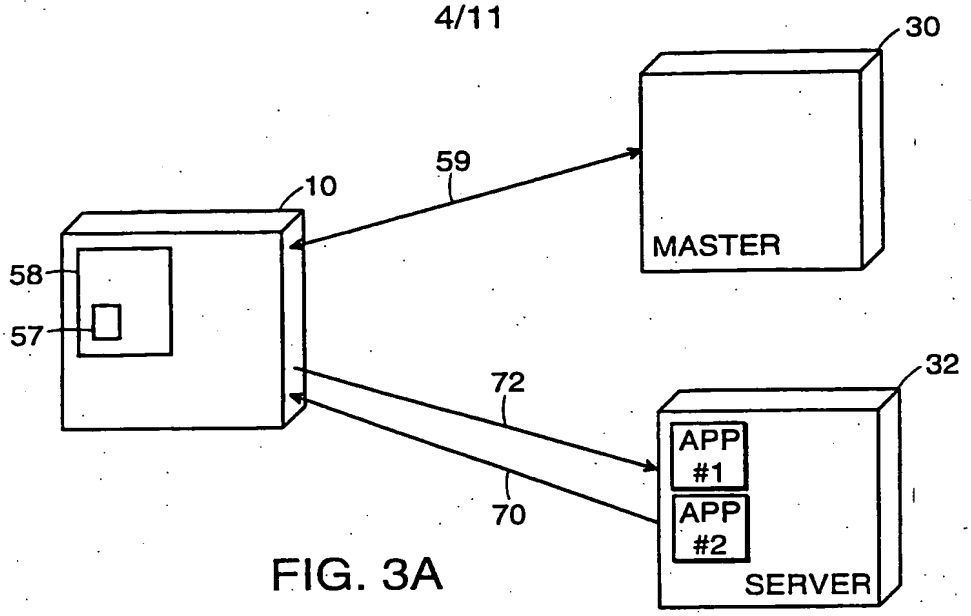

30
59
10
MASTER
58
57
72
32
APP #1
APP #2
70
SERVER

FIG. 3A

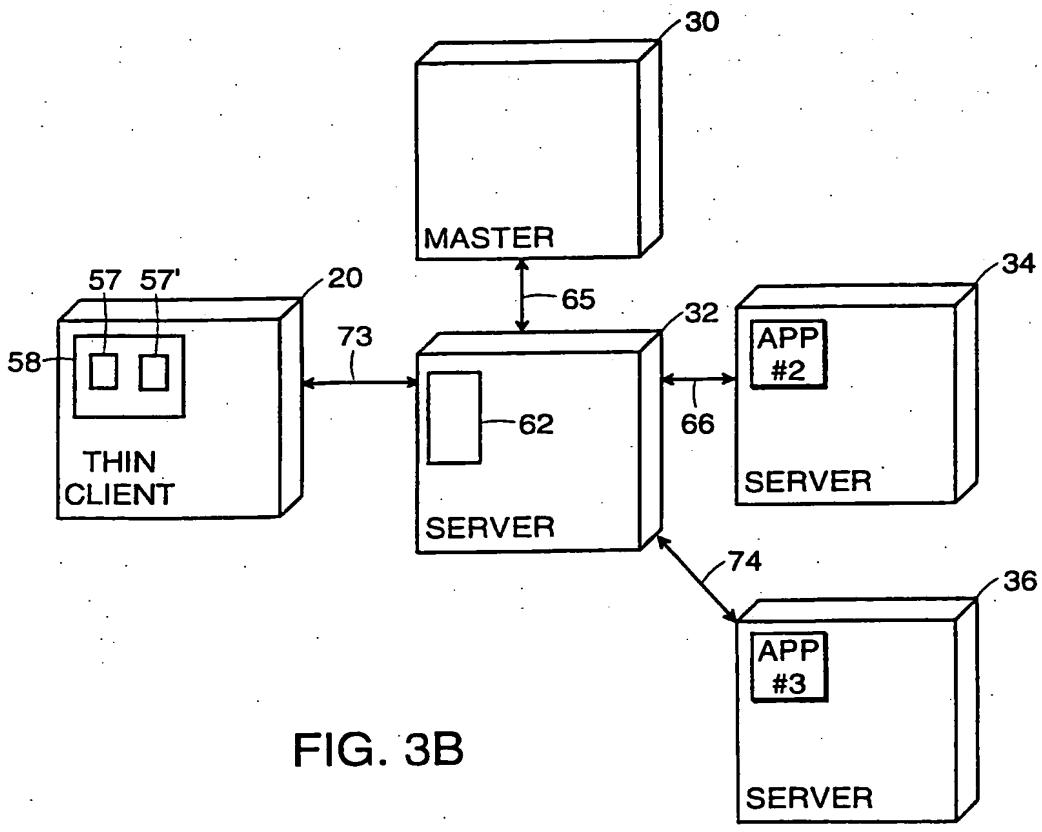

30
MASTER
57 57'
20
65
34
73
32
APP #2
58
62
66
THIN CLIENT
SERVER
SERVER
74
36
APP #3
SERVER

FIG. 3B

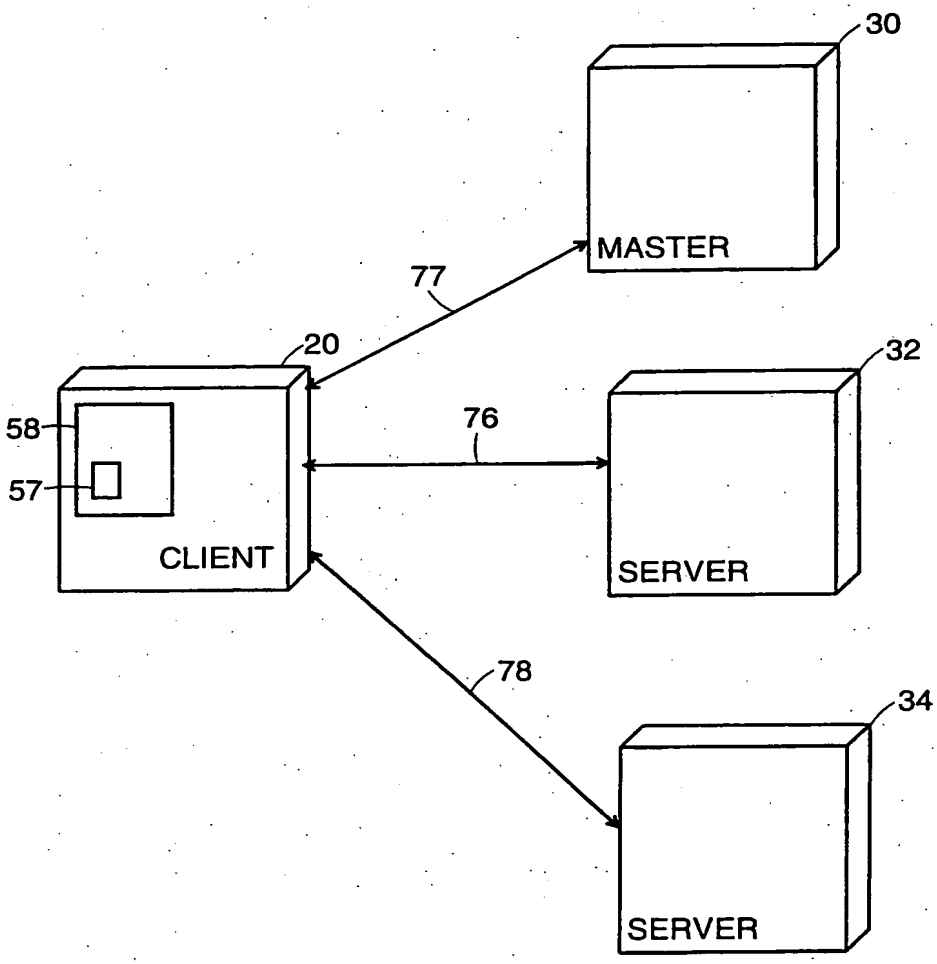


FIG. 3C

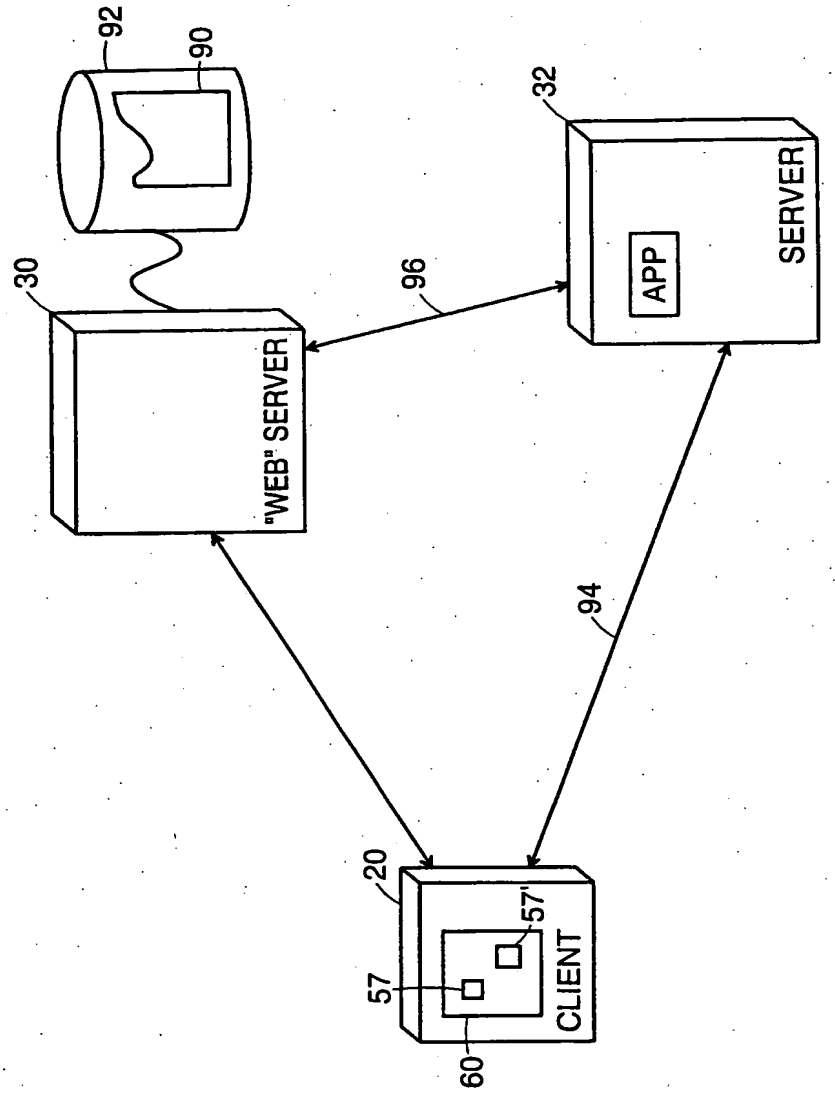


FIG. 3D

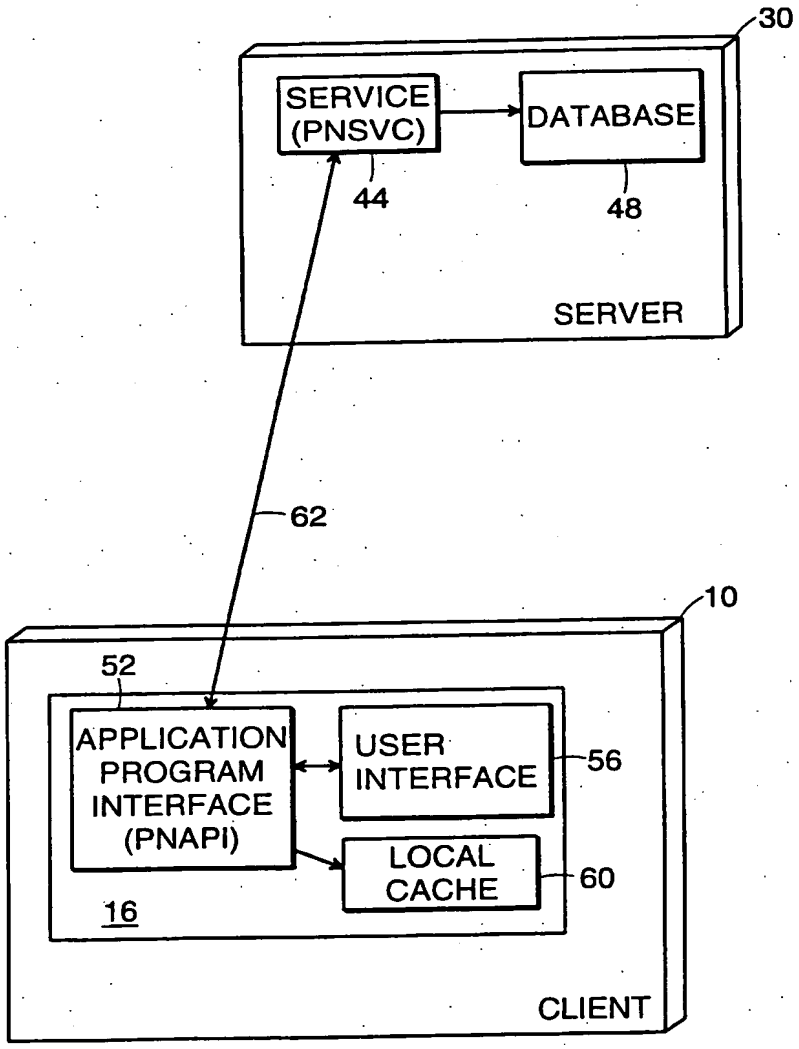

30
SERVICE (PNSVC)
DATABASE
44
48
SERVER
62
10
52
APPLICATION PROGRAM INTERFACE (PNAPI)
USER INTERFACE
56
LOCAL CACHE
60
16
CLIENT

FIG. 4

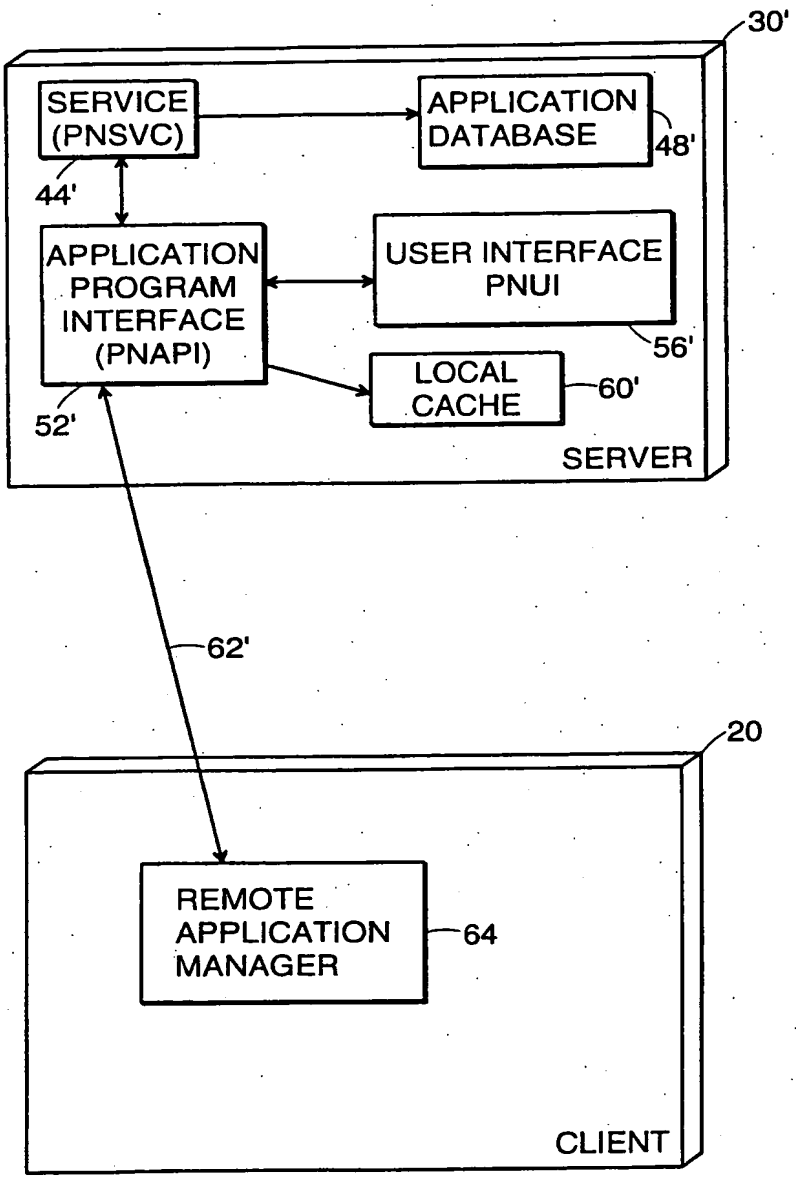

30'
SERVICE (PNSVC)
APPLICATION DATABASE
48'
44'
APPLICATION PROGRAM INTERFACE (PNAPI)
USER INTERFACE PNUI
56'
LOCAL CACHE
60'
52'
SERVER
62'
20
REMOTE APPLICATION MANAGER
64
CLIENT

FIG. 5

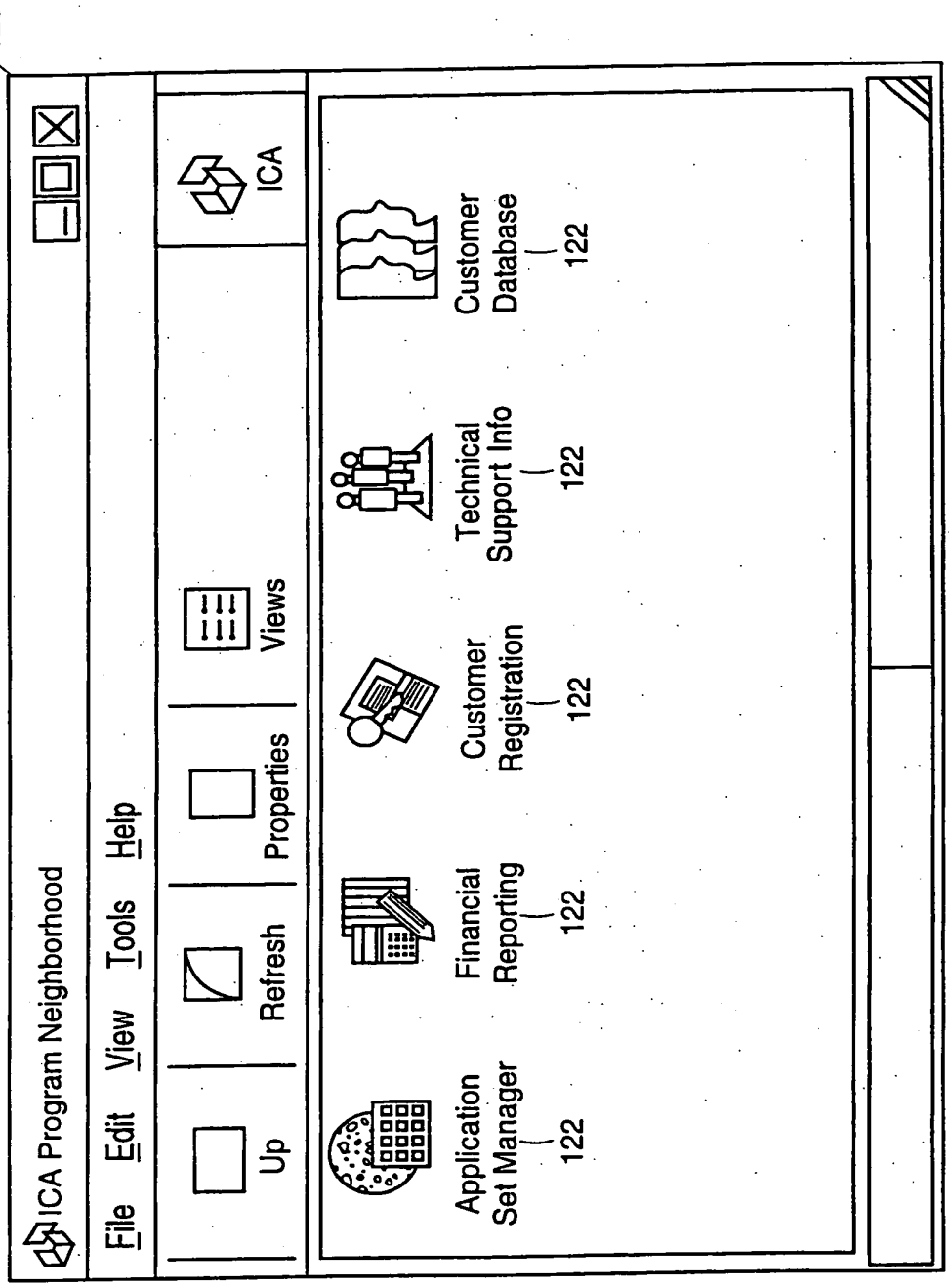

120
ICA Program Neighborhood
File Edit View Tools Help
Up
Refresh
Properties
Views
ICA
Application Set Manager
122
Financial Reporting
122
Customer Registration
122
Technical Support Info
122
Customer Database
122

FIG. 6A

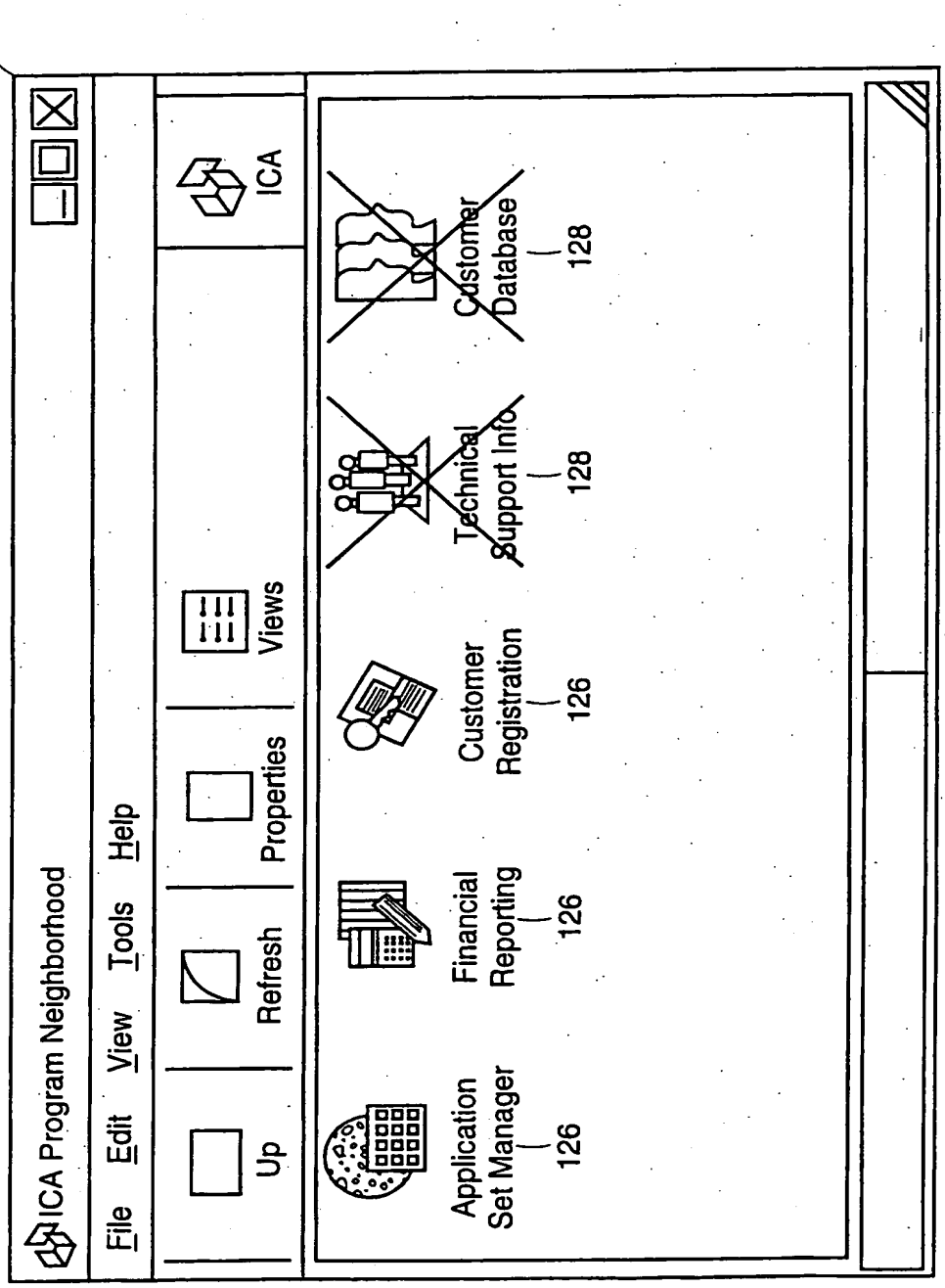

124
ICA Program Neighborhood
File Edit View Tools Help
Up
Refresh
Properties
Views
ICA
Application Set Manager
126
Financial Reporting
126
Customer Registration
126
Technical Support Info
128
Customer Database
128

FIG. 6B

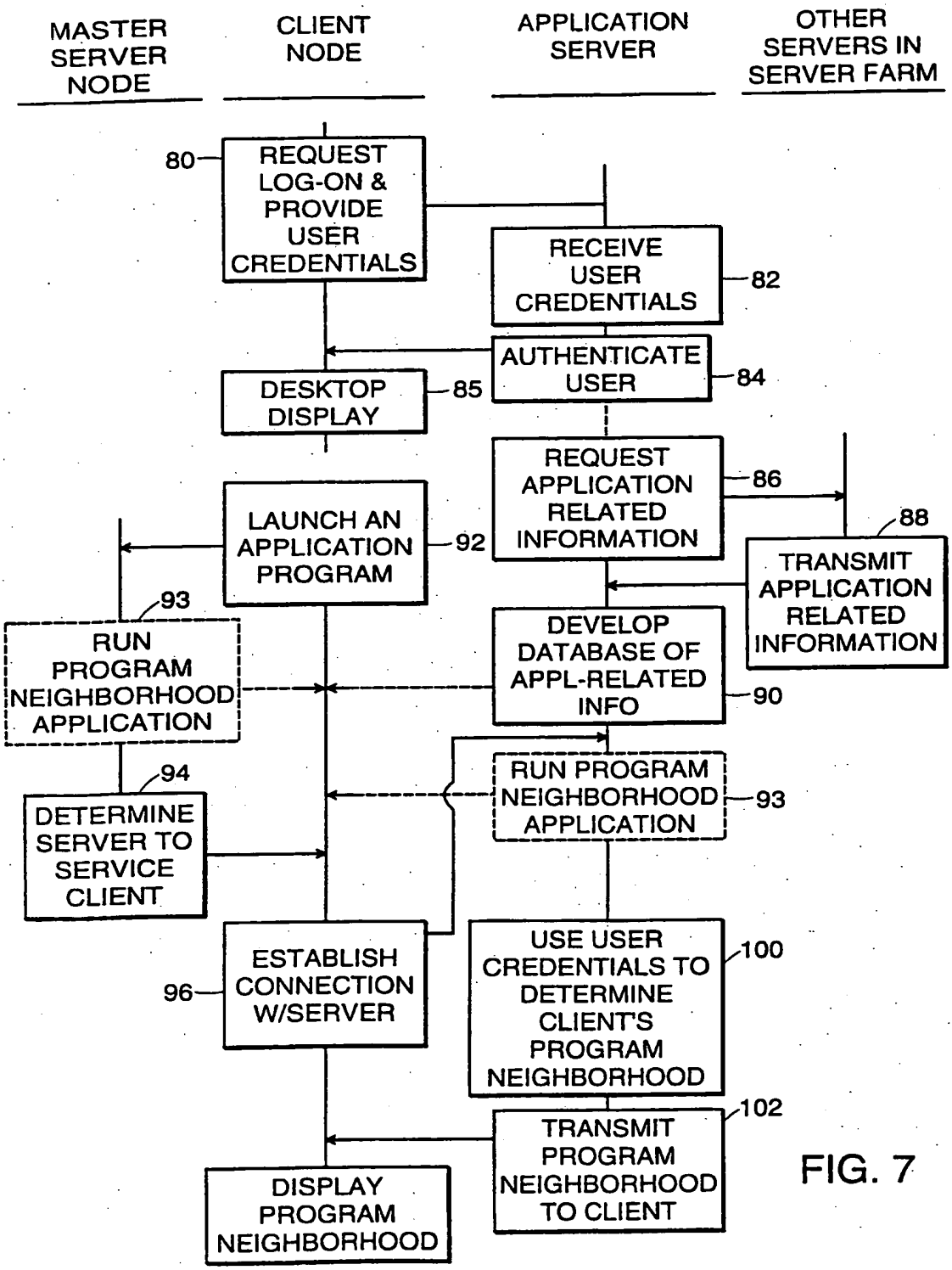

MASTER SERVER NODE
CLIENT NODE
APPLICATION SERVER
OTHER SERVERS IN SERVER FARM
80
REQUEST LOG-ON & PROVIDE USER CREDENTIALS
RECEIVE USER CREDENTIALS
82
AUTHENTICATE USER
84
DESKTOP DISPLAY
85
REQUEST APPLICATION RELATED INFORMATION
86
88
TRANSMIT APPLICATION RELATED INFORMATION
LAUNCH AN APPLICATION PROGRAM
92
93
RUN PROGRAM NEIGHBORHOOD APPLICATION
DEVELOP DATABASE OF APPL-RELATED INFO
90
94
DETERMINE SERVER TO SERVICE CLIENT
RUN PROGRAM NEIGHBORHOOD APPLICATION
93
ESTABLISH CONNECTION W/SERVER
96
USE USER CREDENTIALS TO DETERMINE CLIENT'S PROGRAM NEIGHBORHOOD
100
TRANSMIT PROGRAM NEIGHBORHOOD TO CLIENT
102
DISPLAY PROGRAM NEIGHBORHOOD

FIG. 7

INTERNATIONAL SEARCH REPORT

Inter. nal Application No
PCT/US 99/30827

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F9/46

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X Y A	US 5 689 708 A (REGNIER BARBARA ANN ET AL) 18 November 1997 (1997-11-18) the whole document — -/—	1,12,17, 22,27, 32,36 2-6, 8-11,14, 15, 18-20, 23-26, 28-31, 33-35, 37-40 7,21

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
24 May 2000	14/06/2000

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31–70) 340–2040, Tx. 31 651 epo nl, Fax: (+31–70) 340–3016	Ecolivet, S.

1

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Intern. ..al Application No
PCT/US 99/30827

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5 729 734 A (PARKER ROBERT D ET AL) 17 March 1998 (1998-03-17) abstract; figures 2A,2B,4,5,6	9,30
Y	TOM YAGER: "OS Paradise" BYTE,November 1995 (1995-11), pages 81-98, XP002138588 page 82, right-hand column, line 13 - line 26	33,34
Y	US 5 737 592 A (SRINIVASAN VENKATACHARY ET AL) 7 April 1998 (1998-04-07) abstract column 1, line 22 - line 57 column 2, line 30 - line 50 column 5, line 37 - line 48	28,29,31
A	WO 97 18635 A (IBM UK ;IBM (US)) 22 May 1997 (1997-05-22) abstract page 2, line 41 - line 42 page 3, line 26 - line 39 claims 7-9	

1

INTERNATIONAL SEARCH REPORT

Intern. nal Application No
PCT/US 99/30827

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
Y	B. CLIFFORD NEUMAN: "PROXY-BASED AUTHORIZATION AND ACCOUNTING FOR DISTRIBUTED SYSTEMS" PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON DISTRIBUTED COMPUTING SYSTEMS,US,LOS ALAMITOS, IEEE COMP. SOC. PRESS, vol. CONF. 13, 1993, pages 283-291, XP000399398 ISBN: 0-8186-3770-6	4-6,8
A	page 283, right-hand column, line 25 - line 28, footnote page 285, left-hand column, paragraph 3.2 -page 286, right-hand column, line 19; figure 3	7
X	BRIDGET ALLISON, ROBERT HAWLEY, ANDREA BORR, MARK MUHLESTEIN, DAVID HITZ: "File System Security: Secure Network Data Sharing for NT and UNIX" INTERNET DOCUMENT, 'Online! 1998, XP002138587 Retrieved from the Internet: <URL:http://www.netapp.com/tech_library/1003.html> 'retrieved on 2000-05-22! abstract page 2, line 10 - line 12 page 4, line 17 -page 5, last line page 13, line 11 - line 15	1,12,13, 16,17, 22,36
Y	AU 53031 98 A (MILLS DUDLEY JOHN) 27 August 1998 (1998-08-27)	10,14, 15, 18-20, 23,24, 35,37-40
A	abstract	21
A	page 2, line 19 - line 26 page 4, line 24 - line 29 page 7, line 4 - line 24 page 8, line 28 - last line claims 7-12	28
Y	US 5 826 027 A (PERRY WERNER KURT ET AL) 20 October 1998 (1998-10-20)	2,3,11
A	column 2, line 59 -column 3, line 13 column 9, line 16 - line 65	21
Y	US 5 794 207 A (WALKER ASSET MANAGEMENT LIMITED PARTNERSHIP) 11 August 1998 (1998-08-11) page 9, line 60 - line 67 column 15, line 3 - line 5 column 15, line 46 - line 52 column 24, line 23 -column 26, line 11	25,26

-/--

1

Form PCT/ISA/210 (continuation of second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

information on patent family members

Inter... ...al Application No
PCT/US 99/30827

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5689708 A	18-11-1997	NONE	
AU 5303198 A	27-08-1998	NONE	
US 5826027 A	20-10-1998	NONE	
US 5794207 A	11-08-1998	AU 4247997 A	26-03-1998
		BR 9713193 A	21-12-1999
		CN 1244270 A	09-02-2000
		EP 0954817 A	10-11-1999
		WO 9810361 A	12-03-1998
		US 6041308 A	21-03-2000
US 5729734 A	17-03-1998	NONE	
US 5737592 A	07-04-1998	NONE	
WO 9718635 A	22-05-1997	CN 1202291 A	16-12-1998
		CZ 9801482 A	11-08-1999
		EP 0861551 A	02-09-1998
		HU 9900026 A	28-04-1999
		JP 11500248 T	06-01-1999
		PL 326670 A	12-10-1998